METHOD AND SYSTEM FOR RESTORING DATA

BACKGROUND

1. Technical Field

[0001] The present invention relates to restoring data. More particularly, the invention concerns restoring data and permitting access to the data while the data is being restored.

2. Description of Related Art

[0002] In modern computing systems, data is often backed up to provide protection from data loss if the original data becomes corrupted, and/or to archive the data. Magnetic or optical tape is often used for backing up data. Because the data is stored sequentially on the tape, restoring the data takes the amount of time required to read the tape from the beginning to the end of the data.

[0003] An application program may require access to data that has been backed up onto tape. However, before the application can use the data, the application must wait for the data to be retrieved from the backup. More specifically, when an application requests a file which must be restored from tape, the application must wait for the time required to restore the entire file. The delay is linearly proportional to the size of the file, and is exacerbated with large files. This delay is present when data is restored from a sequential data format, such as tape, to a direct access format such as a magnetic disk, and more generally is present whenever data is restored from one format to another, for example when compressed archived data is restored to an uncompressed format.

[0004] Although an application may require information from only a portion of a large file, with previously known techniques the application cannot be allowed access to the desired portion of the file until the entire file has been restored. This is the case because no facility exists to prevent the application from attempting to access the unrestored portions of the file, which would result in an error. Consequently, known

techniques for restoring data from backup are inadequate for allowing speedy access to the data.

SUMMARY

[0005] One aspect of the invention is a method for restoring data. An example of the method includes receiving a request for at least a portion of the data. This example also includes creating a directory entry for the data in a virtual file system, and allocating storage space for the data. This example further includes initializing a block virtualization indicator to a value indicating that the data is not available. This example additionally includes writing a subset of the data to the storage space, and changing the block virtualization indicator to a value indicating that the data is available.

[0006] Other aspects of the invention are described in the sections below, and include, for example, a computing system, and a signal bearing medium tangibly embodying a program of machine-readable instructions executable by a digital processing apparatus to perform a method for restoring data.

[0007] The invention provides a number of advantages. For example, some examples of the invention advantageously permit access to a portion of a file without first waiting for the entire file to be restored. The invention also provides a number of other advantages and benefits, which should be apparent from the following description.

BRIEF DESCRIPTION OF THE DRAWINGS

[0008] FIG. 1 is a block diagram of the hardware components and interconnections of a computing system in accordance with an example of the invention.

[0009] FIG. 2 is a block diagram of the hardware components and interconnections of a computing apparatus in accordance with an example of the invention.

[0010] FIG. 3 is an example of a signal-bearing medium in accordance an example of the invention.

[0011] FIGS. 4A-4C are a flowchart of an operational sequence for restoring data in accordance with an example of the invention.

DETAILED DESCRIPTION

[0012] The nature, objectives, and advantages of the invention will become more apparent to those skilled in the art after considering the following detailed description in connection with the accompanying drawings.

I. HARDWARE COMPONENTS AND INTERCONNECTIONS

[0013] One aspect of the invention is a computing system for restoring data. As an example, the computing system may be embodied by all, or portions of, the computing system 100 shown in FIG. 1. The computing system 100 may include a host 102 a file virtualization meta data server 104, a storage virtualization engine SAN volume controller 106, a disk device 108, a backup device 110, and may also include a backup server 112, all of which may be coupled together via a SAN (Storage Area Network) 114. Alternatively, a SAN 114 need not be utilized, and the components of the computing system 100 may be coupled directly or through other types of networks.

[0014] The host 102 may include an application program 115, a virtual file system client 116, and a device driver 118. As an example, the host 102 system may run one or more application programs that require storing and retrieving data. The host 102 could be implemented, for example, with a computer running the AIX operating system, a Windows 2000 server , or generally any Intel based PC server system, and in a specific example, may be an IBM® P series 304 server.

[0015] The storage virtualization engine SAN volume controller 106 is used to implement indirect addressing in a virtual file system. With indirect addressing, the address associated with data is not stored with the data. Rather, with a virtual file system (virtual storage), the address is an indirect pointer to the data, and there is a mapping function between the physical and logical addresses. With a virtual file system, the data and the corresponding pointers are stored in different locations. For example, a metadata server could be used for storing the pointers.

[0016] The storage virtualization engine SAN volume controller 106 could be implemented, for example, with an IBM TotalStorage™ SAN Volume Controller, or an

IBM TotalStorage™ San Integration Server. The file virtualization meta data server 104 could be implemented, for example, with an IBM TotalStorage™ SAN File System (based on IBM Storage Tank™ technology). As an example, the disk device 108 could be implemented with an IBM FAStT 900 disk storage system, and the backup device 110 could be an IBM Virtual Tape Storage server (VTS). However, any suitable direct access storage could be used for the disk device 108, any suitable sequential backup storage could be used to implement the backup device 110, and any suitable computing devices could be used to implement the host 102, the file virtualization meta data server 104, the storage virtualization engine SAN volume controller 106, and the backup server 112.

[0017] In an alternative example, the disk device 108 could be coupled directly to the storage virtualization engine SAN volume controller 106, instead of being coupled to the SAN 114. Generally, data movement for restoring data could be through the backup server 112, or through other storage virtualization, or could be via direct device to device transfer. If included, the backup server 112 provides the capability for direct backup from the disk device 108 to the backup device 110.

[0018] An exemplary computing apparatus 200 is shown in FIG. 2. As an example, the host 102, the file virtualization meta data server 104, the storage virtualization engine SAN volume controller106, the backup server 112, and any other computing device in the computing system 100 could be implemented with an example of the computing apparatus 200. The computing apparatus 200 includes a processor 202 (which may be called a processing device), and in some examples could have more than one processor 202. As an example, the processor may be a PowerPC RISC processor, available from International Business Machines Corporation, or a processor manufactured by Intel Corporation. The processor 202 may run any suitable operating system, for example, Windows 2000, AIX, Solaris™, Linux, UNIX, or HP-UX™. The computing apparatus 200 may be implemented on any suitable computing device, for example a personal computer, a workstation, a mainframe computer, or a supercomputer. The computing apparatus 200 also includes a storage 204, a network interface 206, and an input/output 208, which are all coupled to the processor 202. The storage 204 may

include a primary memory 210, which for example, may be RAM, and a non volatile memory 212. The non-volatile memory 212 could be, for example, a hard disk drive, a drive for reading and writing from optical or magneto-optical media, a tape drive, non-volatile RAM (NVRAM), or any other suitable type of storage. The storage 204 may be used to store data and application programs and/or other programming instructions executed by the processor. The application programs could generally be any suitable applications. The network interface 206 may provide access to any suitable wired or wireless network.

II. OPERATION

[0019] In addition to the hardware embodiments described above, other aspects of the invention concern a method for restoring data.

A. Signal-Bearing Media

[0020] In the context of FIGS. 1 and 2, the method aspects of the invention may be implemented, for example, by having the host 102, and in some examples, also one or more of the file virtualization meta data server 104, the storage virtualization engine SAN volume controller 106, and the backup server 112, and possibly also the disk device 108 and/or the backup device 110, execute a sequence of machine-readable instructions, which can also be referred to as code. These instructions may reside in various types of signal-bearing media. In this respect, some aspects of the present invention concern a programmed product, comprising a signal-bearing medium or signal-bearing media tangibly embodying a program of machine-readable instructions executable by a digital processing apparatus to perform a method for restoring data.

[0021] This signal-bearing medium may comprise, for example, primary memory 210 and/or non-volatile memory 212. Alternatively, the instructions may be embodied in a signal-bearing medium such as the optical data storage disc 300 shown in FIG. 3. The optical disc can be any type of signal bearing disc or disk, for example, a CD-ROM, CD-R, CD-RW, WORM, DVD-R, DVD+R, DVD-RW, or DVD+RW.

Additionally, whether contained in the computing system 100, or elsewhere, the instructions may be stored on any of a variety of machine-readable data storage mediums or media, which may include, for example, a "hard drive", a RAID array, a magnetic data storage diskette (such as a floppy disk), magnetic tape, digital optical tape, RAM, ROM, EPROM, EEPROM, flash memory, programmable logic, any other type of firmware, magneto-optical storage, paper punch cards, or any other suitable signal-bearing media including transmission media such as digital and/or analog communications links, which may be electrical, optical, and/or wireless. For example, in some embodiments the instructions or code may be accessible from a file server over a network, or from other transmission media, and the signal bearing media embodying the instructions or code may comprise a transmission media, such as a network transmission line, wireless transmission media, signals propagating through space, radio waves, and/or infrared signals. Additionally, the signal bearing media may be implemented in hardware logic, for example, an integrated circuit chip, a Programmable Gate Array (PGA), an Application Specific Integrated Circuit (ASIC). As an example, the machine-readable instructions may comprise software object code, compiled from a language such as "C++".

B. Overall Sequence of Operation

[0022] For ease of explanation, but without any intended limitation, exemplary method aspects of the invention are described with reference to the computing system 100 described above and shown in FIG. 1. An example of the method aspect of the present invention is illustrated in FIGS. 4A-4C, which shows a sequence 400 for a method for restoring data. In some examples of the invention, the data may be restored from tape. However, the invention could be beneficially utilized in any environment where the physical storage attributes of storage change, for example, from sequential to direct access, or from direct access to archive. More generally, the invention could be beneficially utilized when the data format has been changed such that the data can not be accessed directly and must be restored to an accessible format.

[0023] The operations of the sequence 400 may be performed by the host 102, which may be running an application program 115 that requests the data. The host 102 may also run a backup and restore program. Alternatively, the operations of the sequence 400, may be performed by the host 102 in conjunction with suitably modified versions of the storage virtualization engine 106, the file virtualization meta data server 104, and/or a restore program, and in the case where blocks of the data are restored in an optimized order, a modified backup program.

[0024] Referring to FIG. 4A, the sequence 400 may include, and may begin with, operation 402, which comprises backing up data. Small or large amounts of data may be backed up. For example, hundreds of gigabytes of data could be backed up. As an example, the data may be backed up onto tape. Disk storage is typically organized in blocks of data, which may be large or small. For example, a block could be 512 Bytes or 2048 Bytes. The operation of backing up the data may include storing information identifying the storage locations of each of a plurality of blocks of the data. Herein the term "data" includes any type of data, for example, files and/or raw data.

[0025] Backing up the data may further comprise storing metadata with the data. The metadata may include information concerning access characteristics of blocks of the data, which permits optimizing the order in which data blocks are restored. Optimizing the order in which data blocks are restored can improve performance, in comparison with restoring data blocks sequentially, which can produce uneven performance. The access characteristics metadata may be used in embodiments in which the data is retrieved in an order based on the access characteristics. The data access pattern (access characteristics) metadata may be used to predict the order in which parts of the data are likely to be accessed by the application. The data access patterns may be tracked and the metadata may be stored with the data. Alternatively, rather than storing the metadata with the data, the operation of backing up the data may comprise associating metadata that indicates access characteristics of blocks of the data, with the data.

[0026] The sequence 400 may further include operation 404, which comprises receiving a request for at least a portion of the data. For example, an application program

115 may request access to a data file. The sequence 400 may further include operation 406, which comprises creating a directory (or folder) entry for the data in a virtual file system. (Unix and DOS use the term "directory", whereas Macintosh computers and Windows use the term "folder.") The operation of creating a directory entry may include creating a pointer for the data in the virtual file system. The sequence 400 may further include operation 408, which comprises allocating sufficient storage space for the data in a physical storage (which may be a file). The sequence 400 may further include operation 410, which comprises initializing a block virtualization indicator to a value indicating that the data is not available.

[0027] The sequence 400 may further include operation 412, which comprises identifying if an application performs a write that does not require a read/modify/write on a block of the data that has not yet been restored, and if so the sequence 400 may also include operation 414, which comprises marking the block of the data as discarded for purposes of the restore.

[0028] Referring to FIG. 4B, the sequence 400 may further include operation 416, which comprises writing a subset of the data to the storage space. Operation 416 may also include retrieving the data. The sequence 400 may further include operation 418, which comprises changing the block virtualization indicator to a value indicating that the data is available. The virtualization entry is updated to indicate where the data is stored instead of indicating that the data is not available. In some examples, 10% of the blocks, 1 block, or 10,000 blocks may be retrieved and written to the storage space before the block virtualization indicator is changed to indicate that the data is available. The default amount to be retrieved before the block virtualization indicator is changed may be changed.

[0029] The sequence 400 may further include operation 420, which comprises writing an additional subset of the data to the storage space. In other words, the restoration of the remainder of the data continues while the data is made available to the application.

[0030] The sequence 400 may further include operation 422, which comprises identifying portions of the data that have not been written to the storage space. If a request for a part of the data that is written to the storage space is received, the application is allowed to access that part of the data. The sequence 400 may also include operation 424, which comprises receiving a request for a part of the data that at least partially is not written to the storage space. If a request is received for part of the data that at least partially is not written to the storage space, the sequence 400 may also include operation 426, which comprises indicating a busy condition, which may cause an application and/or operating system to enter a wait state. If the requested part of the data has not been restored by the time of the request, a delay will result because the system may wait to respond to the I/O request, or because the system may send a busy message to the requesting program (and the requesting program may then retry the data request).

[0031] Referring to FIG. 4C, the sequence 400 may further include operation 428, which comprises retrieving and writing into the storage space, the requested part of the data that has not been written to the storage space. The sequence 400 may also include operation 430, which comprises responding to the request for the part of the data.

[0032] The sequence 400 may further include operation 432, which comprises writing an additional subset of the data to the storage space. The additional subset of the data may be retrieved starting at a location sequentially after the retrieved data. Alternatively, the additional subset of the data may be retrieved starting at a location wherein data is expected to be requested next. Alternatively, the additional subset of the data may be retrieved by starting at a randomly selected location. The requested part of the data that is not written to the storage space may be retrieved on a priority basis. The operation 432 of retrieving an additional subset of the data may be repeated until all of the data has been retrieved. Also, operations 422 to 432 may be repeated if an additional request for a part of the data that at least partially is not written to the storage space, is received.

[0033] Some examples of the invention may be embodied as a backup/restore program, a virtualization engine, and a tape virtualization controller. An exemplary

embodiment of the invention may be summarized as follows: A file is restored into a storage virtualization engine, and special entries in the virtualization mapping are used to indicate that the data is temporarily unavailable. Once a subset of the file is available, the file is released for application access. If accesses to unavailable portions of the file are detected (by having an unavailable mapping in the virtual device), completion of those particular I/O's is delayed. Thus, an application is permitted to access the portion of the data that has been restored, and waits for data that has not yet been processed. In this manner, partial access to the file can be allowed while preventing incorrect accesses to the unrestored portions. Further optimization of the restore process is possible, to minimize the amount of time an application has to wait for data. The technique of marking data inaccessible in the virtualization engine may also be used to implement block-level Hierarchical Storage Management (HSM), where infrequently accessed blocks are moved to less costly disk, or to tape.

[0034] In instances where the following prerequisites are present, with some examples of the invention it is possible to retrieve blocks of data in an order in which the blocks are expected to be used by the application, to further minimize the application wait time. The prerequisites are:

-Accesses to the data after the restore is predictable in some way.

-It is not prohibitively costly (in elapsed time) to perform the restore in a non-sequential order, such that blocks more likely to be accessed are restored first.

-A block virtualization engine is available, which can recognize which data has not been restored and can introduce a block-level wait for the unavailable data.

-Applications can tolerate such waits.

[0035] If the preceding prerequisites exist, then it is possible to reduce application wait time by restoring data blocks in a particular order. This alternative embodiment may be implemented by storing access metadata with the data when the data is backed up. Alternatively, file access characteristics metadata may be associated with the file after the data is restored.

[0036] An exemplary implementation of the invention may be described as follows. At restore time, a directory entry for a file that is to be retrieved is made in a virtual file system, and space is allocated for the file. The block virtualization is initialized to be equal to a "not present being restored" value. Data is then restored in chunks, randomly (instead of serially) in the order that references are expected. Alternatively, the data may just be restored sequentially in the hope it will be accessed in the same way. Once a sufficient quantity of data has been loaded using the expected access pattern, the application that requested the data is allowed to access the data. The amount of data that is sufficient is the quantity of data required so the application will not be likely to experience a delay when accessing the data. The quantity of data that is sufficient is a function of the access pattern of the data and the data type. If the application reads the file before a block desired by the application has been restored, the read may be held and a priority restore action for the desired block may be placed in the queue of items to be restored. If the application performs a write which does not require a read/modify/write, on a block that has not yet been restored, that block may be marked as discarded for purposes of the restore. Unaccessed blocks may be restored sequentially to give good tape performance.

[0037] The degree to which examples of the invention can improve data access performance is a function of the amount of data to be restored and the amount of data that the application seeks to access. With some examples of the invention, data could be available in 1/100 of the time, for example, that would otherwise be required. The most significant improvement generally occurs when a large amount of data must be restored and when the amount of data needed by the application program 115 is small in relation to the total amount to be restored.

III. OTHER EMBODIMENTS

[0038] While the foregoing disclosure shows a number of illustrative embodiments of the invention, it will be apparent to those skilled in the art that various changes and modifications can be made herein without departing from the scope of the

invention as defined by the appended claims. Furthermore, although elements of the invention may be described or claimed in the singular, the plural is contemplated unless limitation to the singular is explicitly stated.